Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Kern

Internetna naprava za merjenje in analizo porabe električne energije

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Danes nas elektronske naprave na podlagi podatkov, ki so na voljo na spletu, neprestano obveščajo o vsem mogočem. Še najmanj informacij dobimo o stanju večine gospodinjskih aparatov. V okviru naloge izdelajte napravo za merjenje in analizo porabe električne energije, preko katere lahko klasične gospodinjske aparate povežete v internet stvari. Raziščite kakšne in kako zanesljive informacije o delovanju naprave dobite iz meritev in kako lahko z njihovim posredovanjem uporabniku pomagate pri vsakodnevnih opravilih.

Zahvlajujem se mentorju izr. prof. dr. Urošu Lotriču za pomoč pri izdelavi diplomskega dela. Zahvalil bi se še družini, ki mi je v času izdelave diplomskega dela stala ob strani.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IoT	internet of Things	internet stvari
RFID	radio-frequency identification	radiofrekvenčna identifikacija
M2M	machine to machine	komunikacija med napravami
TSDB	time series database	podatkovna baza za shranjevanje časovnih vrst
IR	infrared radiation	infrardeče valovanje
ADC	analog-to-digital converter	analogno-digitalni pretvornik
UART	universal asynchronous receiver/transmitter	univerzalni asinhroni sprejemnik in oddajnik
LCSS	longest common subsequence	najdaljši skupni podniz
DTW	dynamic time warping	dinamično časovno upogibanje
ERP	edit distance with real penalty	Levenshteinova razdalja z realno kaznijo
PAA	piecewise aggregate approximation	aproksimirano kosovno združevanje

Povzetek

Naslov: Internetna naprava za merjenje in analizo porabe električne energije

Avtor: Žiga Kern

Popularnost naprav, povezanih v internet, v zadnjih letih močno narašča. Kljub temu pa v današnjem trgu ne obstaja veliko rešitev za povezovanje obstoječih naprav v internet. V tem diplomskem delu si bomo ogledali trenutno ponudbo povezljivih naprav na trgu in možnosti nadgradnje obstoječih naprav. V nadaljevanju bomo predstavili razvoj naprave, ki nam omogoča takšno nadgradnjo brez posegov v obstoječo napravo. Opisali bomo razvoj strojne opreme za zajem napetosti in toka, izračun podatkov o porabi električne energije in posredovanje teh v računalnik. Predstavili bomo način hranjenja časovnih vrst v računalniku, njihovo obdelavo za prikaz podatkov uporabniku in možnost obveščanja o vzorcih porabe električne energije. Ogledali si bomo tudi nekaj najpopularnejših algoritmov za zmanjševanje dimenzionalnosti podatkov in prepoznavanje vzorcev v časovnih vrstah za potrebe obveščanja.

Ključne besede: nadgradnja, naprava, internet stvari, električna poraba, analiza.

Abstract

Title: Internet device for measurement and analysis of electrical consumption

Author: Žiga Kern

The popularity of devices connected to the Internet has been rising steadily in recent years. Nevertheless, in today's market, there are not many solutions for connecting existing devices to the Internet. In this thesis, we will present an overview of available devices and the possibilities of upgrading the existing devices. Next, we will show how we developed a device that allows us to upgrade an existing one without the need to modify it. We will describe the hardware needed for measuring voltage and current, describe the procedure for calculating power consumption and the method of transmitting data to the computer. We will show how to store time series on your computer, process the series for displaying data to the user and describe how to notify the user about patterns of power consumption. We will also look at some of the most popular algorithms for reducing the dimensionality of data and identifying patterns in time series for the purpose of notifying the user.

Keywords: upgrading, device, internet of things, power consumption, analysis.

Poglavje 1

Uvod

Internet stvari je omrežje med seboj povezanih fizičnih naprav, senzorjev, stavb in drugih naprav, katerega velikost v zadnjih letih hitro narašča. Po nekaterih virih [34], naj bi bilo v letu 2017 aktivnih kar 8,4 milijard naprav. Naprave, povezane v internet, se uporabljajo na zelo različnih področjih, od industrije pri nadzorovanju proizvodnje, pri nadzorovanju javne infrastrukture do upravljanja naprav v domovih.

Najbolj zanimivo področje za potrošnike, pa je prav gotovo koncept pametnega doma. Po raziskavi [26] iz leta 2016 naj bi bilo teh v letu 2015 17,9 milijonov. Najbolj popularne naprave na tem področju so pametni termometri, varnostni sistemi, pametne žarnice in več-prostorsko pametno ozvočenje.

Z rastočim trgom se pojavlja tudi veliko različnih rešitev in ponudb, toda visoka cena teh preprečuje nadgradnjo celotnega doma in vseh naprav povprečnemu potrošniku. Na voljo je tako rekoč vse od že znanih pametnih televizij, pametnih hladilnikov, pralnih strojev do pametnih košev za odpadke. V tej diplomski nalogi preučimo možnosti nadgradnje obstoječih naprav, brez potrebe po nakupu novih, in primer naprave za nadgradnjo tudi zasnujemo.

V 2 poglavju si podrobno ogledamo področje IoT (ang. Internet of Things) in IoT naprav, ki so trenutno ponujene na trgu. V drugem delu poglavja 2.2 pa preučimo možnosti nadgradnje obstoječih naprav.

V 3 poglavju predstavljamo primer naprave za nadgradnjo naprav, kot

so sušilni, pralni stroj, mikrovalovne in navadne pečice, z merjenjem njihove porabe, ki zna uporabnika obveščati o raznih vzorcih električne porabe.

V podpoglavju 3.1 predstavimo osnovni koncept merjenja porabe naprav, ki uporabljajo izmenično energijo, sledi podroben opis posameznih delov strojne opreme, uporabljene za izdelavo naprave, shema tiskanega vezja in ogled končne sestavljene naprave.

V naslednjem podpoglavju 3.2 si ogledamo programsko opremo, ki na podlagi meritev senzorjev izračunava porabo naprave in jo posreduje glavnemu procesorju preko naprave UART (ang. universal asynchronous receiver/transmitter). Sledi predstavitev načina hranjenja podatkov časovne vrste in njihovega vnosa v podatkovno bazo za časovne vrste (ang. time series database, TSDB). Ogledamo si še možne algoritme za zmanjševanje dimenzionalnosti podatkov in algoritme za ujemanje vzorcev ter opišemo zakaj smo se odločili za izbrane algoritme. Na koncu poglavja pa predstavimo še uporabniški vmesnik in obveščanje preko spletnih push obvestil.

V 4 poglavju analiziramo uspešnost zaznavanja vzorcev zasnovane naprave in jo testiramo v gospodinjstvu, na zadnje pa sledi še zaključek v katerem povzamemo naše ugotovitve.

Poglavje 2

Pregled področja

2.1 IoT

Koncept povezanih naprav izvira iz 70-ih let, prva IoT naprava pa naj bi po nekaterih virih [23, 35] izvirala iz zgodnjih 80-ih let iz univerze Carnegie Melon. Ta naj bi bila modificiran avtomat za Coca-Colo, ki je uporabniku sporočal, če je na voljo ohlajena pijača.

Izraz IoT se je pojavil šele skoraj dve desetletji kasneje, ko ga je leta 1999 uporabil Kevin Ashton, popularen pa je glede na podatke Google Trends [33] postal šele v zadnjih letih. Ashtonova ideja je temeljila predvsem na tehnologiji radiofrekvenčne identifikacije (ang. radio-frequency identification, RFID), kar je mnogo drugače kot danes, ko se stvari oziroma naprave povezujejo v internet.

Ena izmed najbolj znanih IoT naprav je prav gotovo v internet povezan opekač za kruh, ki je bil predstavljen leta 1990 [30], leta 2000 pa je na trg prišel tudi pametni hladilnik podjetja LG. Danes je tako na trgu na voljo veliko različnih naprav IoT vedno večih proizvajalcev. Na voljo so naprave kot na primer pametni pralni stroji, pametne ključavnice, stikala, sistemi za ozvočenje, kavni avtomati in pametne televizije. Kupimo lahko tudi sisteme za osvetljavo in nadziranje centralnega ogrevanja, kjer sta med bolj znanimi Phillipsov Hue in thermostat podjetja Nest, ki med drugim ponuja tudi pa-

metne detektorje dima in varnostne nadzorne sisteme.

Večino naprav lahko uporabljamo samostojno, nekatere pa lahko povezujemo tudi v sisteme s centralnim kontrolerjem. Med najpopularnejšimi kontrolerji sta Samsungov SmartThing Hub in Vera, preko katerih lahko potem opravljamo z več napravami hkrati, kakšna opravila pa lahko tudi avtomatiziramo. S kontrolerji lahko povežemo le naprave, ki podpirajo enega izmed podprtih protokolov, ki pa jih je z vsakim letom več. Najbolj razširjeni so prav gotovo WiFi, BlueTooth in mobilni protokoli, toda ti mnogokrat niso primerni za IoT naprave zaradi relativno velike porabe energije ali omejitve števila naprav. Za premostitev težav je bilo razvitih veliko različnih protokolov kot so na primer Z-Wave, Zigbee, 6LowPAN in Thread [20]. Med protokole spada tudi sporočilni protokol MQTT, ki je bil razvit za komunikacijo med napravami (angl. machine to machine, M2M) in je namenjen za uporabo na vrhu protokola TCP/IP za prenašanje majhnih sporočil.

Poleg fizičnih kontrolerjev obstajajo tudi oblačne storitve, ki so se pojavile v zadnjih letih in opravljajo podobno nalogo. Tukaj je potrebno omeniti Microsoftov Azure IoT Suite [11], Googlov Cloud IoT [12] in Amazonovo storitev IoT [9].

Zgoraj omenjene pametne naprave mnogokrat niso najcenejše, zato nakup teh ni vedno smiseln, še posebej kadar že imamo delujoče naprave brez povezljivosti. Da se izognemo nakupu novih naprav, lahko te tudi nadgradimo. Nadgradnja naprav je lahko dokaj preprosta z nakupom namenske naprave, lahko pa postane zelo zapletena, odvisno od naprave, ki jo želimo nadgraditi. Več o možnostih in načinih nadgradnje bomo povedali v naslednjem podpoglavju.

IoT naprave lahko izdelamo tudi sami. Za ta namen so na trgu na voljo razne naprave in komunikacijski moduli, kot so na primer Arduino Yun, Raspberry Pi in komunikacijski modul Z-Wave. Poleg naprav obstajajo tudi odprto kodne knjižnice, načrti za izdelavo takšnih naprav in mnogo literature [32, 28].

2.2 Možnosti nadgradnje

Možnosti za nadgradnjo naprav je mnogo, od bolj preprostih, ki jih lahko opravi kdorkoli, do kompleksnejših, ki zahtevajo veliko vloženega časa. Pred nadgradnjo naprav se moramo sprva vprašati katere funkcionalnosti bi pri napravi oddaljeno upravljali oziroma katere so potrebne, če bi želeli opravilo, ki ga pogosto opravljamo, avtomatizirati. Funkcionalnost kot oddaljeno vklapljanje pomivalnega stroja ni smiselna, saj moramo vanj pred tem zložiti posodo, medtem ko sporočanje o končanem pranju posode morda je.

Ena izmed najpreprostejših nadgradenj, ki jih lahko opravimo, je prav gotovo uporaba ti. pametnih vtičnic in stikal. S pomočjo teh lahko nadgradimo preproste naprave, kjer ne potrebujemo veliko nadzora nad njimi, saj te vtičnice omogočajo le oddaljeno prižiganje in ugašanje priklopljenih naprav. Z njihovo uporabo, bi tako lahko nadzorovali ogrevanje prostora z uporabo električnih radiatorjev ali kaloriferjev, nadzorovali namakanje z vklopom in izklopom črpalke za vodo, opravljali z osvetljavo in podobno.

Nakup sistemov večprostorskega ozvočenja, ki je eden najpopularnejših IoT izdelkov, verjetno ni potreben, saj lahko obstoječe ozvočenje le nadgradimo. Poleg tega, da lahko z nadgradnjo privarčujemo, je ozvočenje, ki ga imamo doma, verjetno še vedno vredno svojega denarja, saj tehnologija v reproduciranju zvoka ne napreduje hitro.

Nadgradnja ozvočenja je dokaj preprosta, saj lahko uporabimo kar Googlov Chromecast Audio, ki omogoča brezžično predvajanje glasbe v več prostorih hkrati. Podobno lahko storimo tudi s televizijo z uporabo enega izmed mnogih HDMI ključkov, s katerim lahko skoraj vsako televizijo spremenimo v pametno.

Za nadgradnjo klimatske naprave, televizije, radijskega sistema ali DVD, Blu-ray predvajalnika in drugih naprav lahko uporabimo njihovo zmožnost kontroliranja preko infrardečih žarkov. Tako, bi si omogočili dostop do vseh funkcij, ki jih naprava omogoča preko daljinskega upravljalca. Za ta namen,

bi lahko izdelati kontroler, ki prevaja infrardeče signale iz ukazov, ki bi jih prejemal preko interneta. Takšna ideja seveda ni nova in med drugimi lahko najdemo tudi odprtokodno rešitev [3].

Še ena možnost nadgradnje naprav je s spremljanjem njihove porabe. S spremljanjem porabe lahko uporabniku sporočamo o raznih vzorcih, ki se pojavljajo v porabi, koliko energije naprava uporablja in v katerih časih dneva je ta najbolj aktivna. Z beleženjem porabe lahko spremljamo na primer čas pečenja hrane v pečici in po določenem času, glede na uporabnikove nastavitve pečico tudi ugasnemo, da se hrana ne zapeče preveč.

Ko želimo upravljati z napravami, ki nimajo možnosti preproste nadgradnje, se posegov vanje ne moremo izogniti. V primeru, da bi na primer želeli nastavljati čas kuhanja in temperaturo posameznih kuhalnih polj štedilnika, bi ga morali nadgraditi, tako da bi zamenjali ali modificirali kontrolno vezje samega štedilnika. Tak način nadgradnje ni primeren za mnoge potrošnike, poleg potrebnega časa se nakup nove pametne naprave verjetno splača bolj kot nadgradnja obstoječe.

Kot primer nadgradnje naprav smo v tej diplomskem delu izbrali merilec porabe energije naprav, s pomočjo katerega lahko spremljamo porabljeno električno energijo in prejemamo obvestila o vzorcih porabe energije, ki nas zanimajo. Izdelan prototip merilca naj bi omogočal preprosto nadgradnjo naprave, podobno kot pametne vtičnice. Več o izdelavi prototipa merilca porabe pa si bomo pogledali v naslednjem poglavju.

Poglavje 3

Implementacija naprave

3.1 Strojna oprema

Merilec porabe električne energije, je sestavljen iz treh glavnih delov. Prvi del naprave je tiskano vezje, ki vsebuje komponente za merjenje napetosti in toka, mikrokrmilnik, napajalnik ter konektor za povezavo z računalnikom Raspberry Pi 3. Drugi del naprave je omenjeni računalnik Raspberry Pi, ki služi za procesiranje in hranjenje podatkov o porabi električne energije, tretji pa je ohišje z vtikačem in vtičnico, ki povezuje prva dva dela skupaj.

V prvem podpoglavju 3.1.1 bomo predstavili princip merjenja porabe električne energije in uporabljeno shemo vezja s katero to dosežemo. V drugem podpoglavju 3.1.2 bomo predstavili računalnik Raspberry Pi 3, komunikacijski protokol za izmenjavo podatkov in potrebno shemo vezja za izvedbo komunikacije. V zadnjem podpoglavju 3.1.3 pa sledi še shema celotnega vezja in opis izdelane naprave.

3.1.1 Merjenje porabe električne energije

Če želimo vedeti koliko električne energije neka naprava porabi v določenem časovnem intervalu, moramo zato poznati moč. Moč za električne naprave izračunamo po enačbi

$$P = UI\cos\phi, \tag{3.1}$$

ki pa se pri enosmernem električnem toku poenostavi v enačbo

$$P = UI, \tag{3.2}$$

saj je fazni zamik v tem primeru enak nič.

Pri napajanju z izmeničnim tokom tok in napetost konstantno nihata z določeno frekvenco, ki je v slovenskem omrežju enaka 50 Hz. Moč v določenem trenutku nam zato ne pomeni veliko in moramo pri računanju moči za izmeničen električen tok upoštevati efektivno (ang. root-mean-square, RMS) vrednost toka in napetosti.

Efektivni vrednosti toka in napetosti za sinusne valovne oblike izračunamo po formuli

$$U_{eff} = \frac{1}{\sqrt{2}} U_{max} \tag{3.3}$$

in

$$I_{eff} = \frac{1}{\sqrt{2}} I_{max}, \tag{3.4}$$

kjer I_{max} in U_{max} predstavljata maksimalni vrednosti toka in napetosti. Za poljubno valovno obliko pa lahko efektivne vrednosti pridobimo z vzorčenjem signala in izračunom efektivne vrednosti po formulah

$$U_{eff} = \sqrt{\sum_{i=0}^{i<n} U_i^2} \tag{3.5}$$

in

$$I_{eff} = \sqrt{\sum_{i=0}^{i<n} I_i^2}, \tag{3.6}$$

kjer n predstavlja skupno število meritev, U_i ter I_i pa vrednosti posamezne meritve.

Navidezno moč naprave, ki je izražena v enotah VA, izračunamo po formuli

$$P_{navidezna} = U_{eff} I_{eff}. \tag{3.7}$$

Uporabnika naprave pa verjetno bolj zanima delovna moč izražena v vatih, ki jo zaračunavajo ponudniki električne energije. Izračun te poteka po formuli

$$P_{delovna} = U_{eff} I_{eff} PF = P_{navidezna} PF, \tag{3.8}$$

kjer se upošteva še faktor moči (PF). Faktor moči pa predstavlja razmerje med delovno in navidezno močjo ter se izračuna po formuli

$$PF = \cos\phi = \frac{P_{delovna}}{P_{navidezna}}. \tag{3.9}$$

O podrobnostih in izračunu jalove moči, ki je tu nismo omenjali, si lahko preberemo v [31] in [24].

Za merjenje toka in napetosti se v elektroniki uporabljajo AD pretvorniki, ki pretvorijo analogni signal v digitalnega. AD pretvorniki so na voljo kot samostojna integrirana vezja ali pa se nahajajo znotraj drugih integriranih vezij.

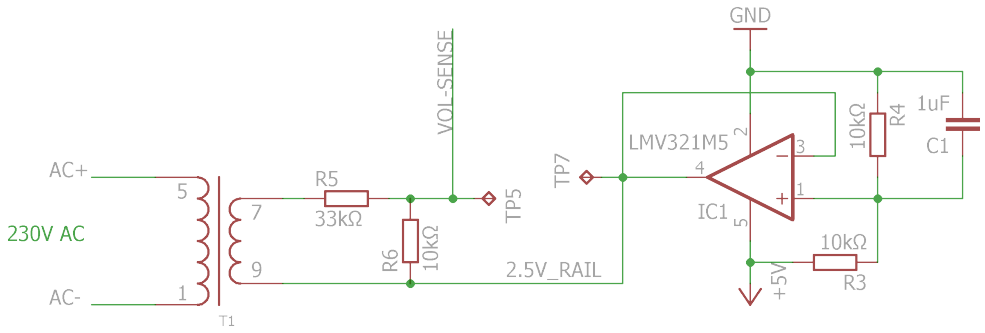
Za merjenje moči izmenične napetosti bi lahko uporabili kar namenska integrirana vezja, kot je na primer Maximovo integrirano vezje 78M6613 AC Power Measurement IC. Ta integrirana vezja se uporabljajo v raznih napravah od napajalnikov do merilcev porabe za preprosto in natančno merjenje električne moči.

V tej diplomski nalogi pa bomo zaradi hitrejšega razvoja prototipa uporabili kar Atmelov mikrokrmilnik ATmega328P z vgrajenim 10-bitnim AD pretvornikom. Kot je razvidno iz podatkovnega lista [5] je delovno območje pretvornika med 0 V in V_{cc}. Ker bomo mikrokrmilnik napajali z napetostjo 5 V, je območje enako $0 - 5$ V.

Če želimo meriti napetost električnega omrežja v Sloveniji, katere efektivna vrednost je enaka $U_{eff} = 230$ V, je potrebno to pretvoriti na prej omenjeno območje. Da to storimo, moramo zmanjšati napetost, ki niha z amplitudo $U_{max} = 325$ V, na napetost, ki bo nihala z amplitudo $U_{max} = 2{,}5$ V, saj je delovno območje AD pretvornika široko le 5 V.

Za zmanjšanje napetosti lahko uporabimo razdelilnik napetosti, sestavljen iz dveh uporov, ki zna napetost tudi zamakniti. Velikost zamika mora v

našem primeru biti 2,5 V, tako da napetost ne bo nihala okrog 0 V, temveč okrog 2,5 V. S tem premikom bomo signal omejili na območje primerno za naš AD pretvornik.

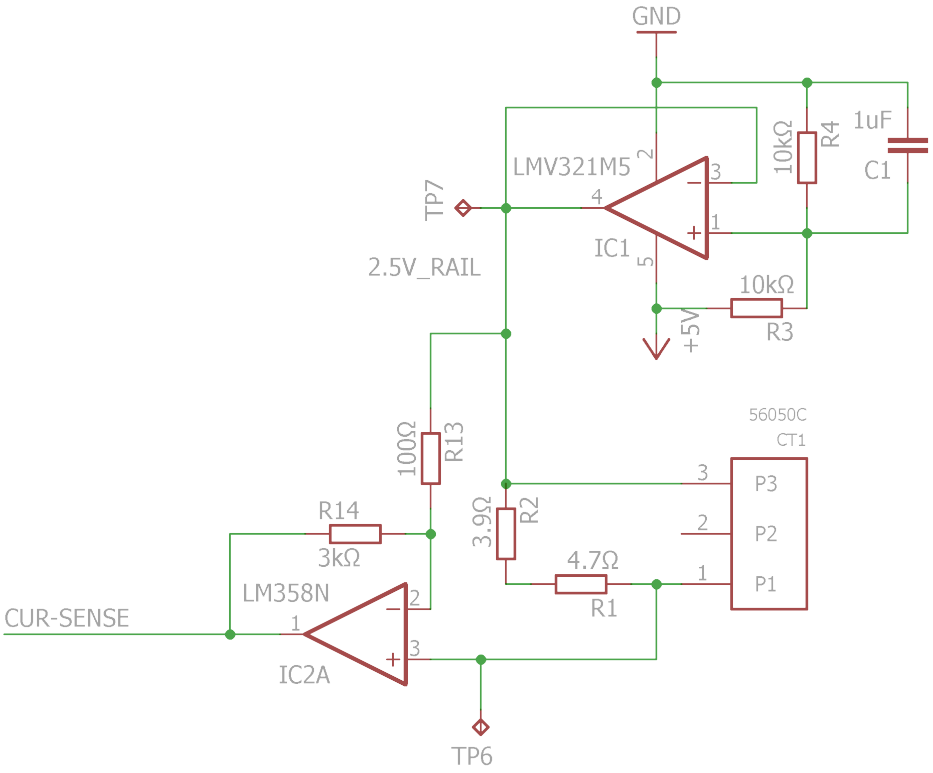


Slika 3.1: Shema vezja za zmanjševanje in zamik napetosti.

Ker želimo napetost zamakniti žal ne moremo uporabiti samo razdelilnika napetosti, saj višja in nižja napetost ne bi bili izolirani. Kot je razvidno iz sheme na sliki 3.1, za ta namen uporabimo poleg razdelilnika še majhen transformator napetosti, ki napetost omrežja pretvori v manjšo in jo hkrati izolira od napetosti, ki jo uporablja naš mikrokrmilnik.

Za premik nato povežemo na en del sekundarne tuljave transformatorja napetost $V_{cc}/2 = 2{,}5$ V, ki jo generira desni del sheme na sliki 3.1, na drugem koncu pa skozi razdelilnik napetosti dobimo naš signal označen kot *VOL-SENSE*, ki ga posredujemo AD pretvorniku.

Pri toku, podobno kot pri napetosti, uporabimo tokovni transformator, ki tok iz primarne tuljave, ki je v našem primeru žica napajalnega kabla naprave, pretvori v manjši tok v sekundarni tuljavi. Da tok pretvorimo v napetost, ki jo lahko AD pretvornik pretvori v digitalni signal, uporabimo upor, povezan med koncema sekundarne tuljave. Kot je razvidno iz podatkovnega lista [1], s tem dobimo pri 10 A toka skozi primarno tuljavo signal z napetostjo, ki niha za 90 mV.



Slika 3.2: Shema vezja za zaznavanje toka.

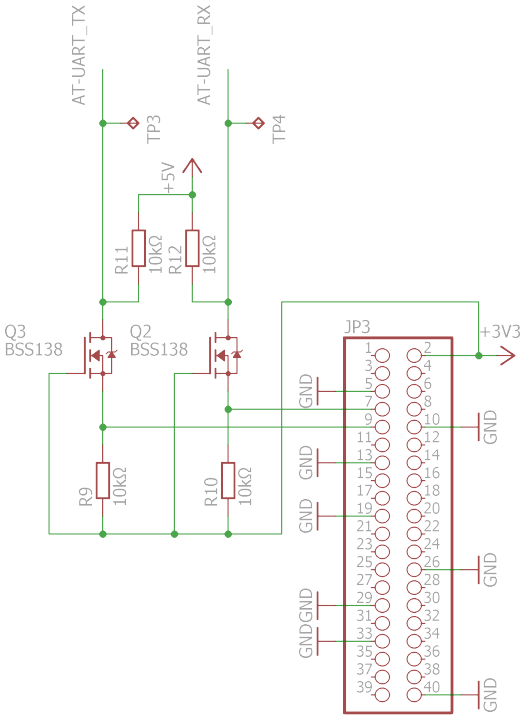
Signal še zamaknemo z napetostjo $V_{cc}/2 = 2{,}5$ V in ga s pomočjo operacijskega ojačevalnika CT1 iz sheme na sliki 3.2 ojačamo. Z ojačanjem dobimo signal, ki pokriva večji delež območja delovanja AD pretvornika in s tem dobimo večjo natančnost zaznavanja toka.

Sedaj, ko so signali zajeti, sledi le še izračun moči. Več o tem in o posredovanju teh podatkov v računalnik Raspberry Pi za hranjenje in procesiranje, si lahko preberemo v poglavju 3.2.1, ki govori o programiranju mikrokrmilnika. Nekaj informacij o uporabljenih shemah vezij lahko najdemo na spletni strani [14].

3.1.2 Raspberry Pi

Raspberry Pi je majhen računalnik velikosti kreditne kartice, prvotno namenjen za uporabo v izobraževanju. Danes se uporablja za veliko različnih namenov, najdemo ga v raznoraznih vgrajenih napravah pa tudi kot nadomestek za namizni računalnik. Tretja verzija računalnika Rasperby Pi, ki smo jo uporabili v okviru te seminarske naloge, vključuje štiri jedrni procesor s hitrostjo 1,2 GHz, en gigabajt pomnilnika, modula za bluetooth in wifi ter čitalec kartic microSD.

Računalnik Raspberry Pi ima poleg priključkov HDMI, Micro-B USB in Ethernet še 40 IO priključkov, preko katerih je povezan s tiskanim vezjem za potrebe komunikacije. Računalnik Raspberry Pi smo v napravi uporabljali kot glavni računalnik za hranjenje in analizo podatkov porabe, vanj pa je priključen še Micro-B USB kabel za napajanje.



Slika 3.3: Shema vezja za spreminjanje nivoja napetosti.

Za komunikacijo med računalnikom Raspberry Pi in mikrokrmilnikom ATmega328P uporabljamo serijsko komunikacijo, med napravama pa se nahaja še vezje za spreminjanje napetosti signala. Vezje, ki ga lahko vidimo na sliki 3.3, je nujno potrebno, saj je napetost signalov mikrokrmilnika in računalnika Raspberry Pi različna in bi pri direktni povezavi med napravama prišlo do uničenja ene izmed naprav. Da se temu izognemo, napetost signala s pomočjo tranzistorjev Q2 in Q3 pretvorimo iz 3,3 V v 5 V in obratno, ter s tem omogočimo komunikacijo.

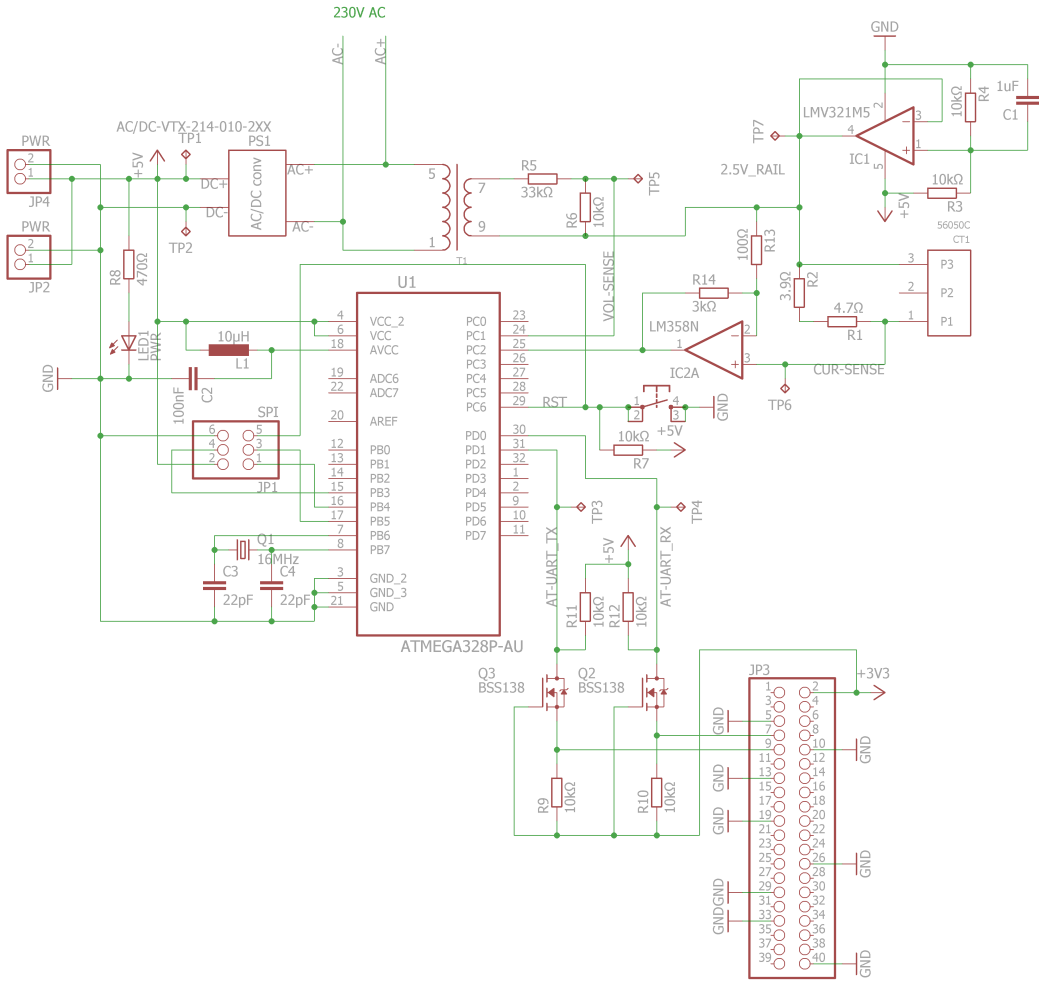
Več o samem Raspberry piju si lahko preberemo na uradni spletni strani [15], več o nameščeni programski opremi pa v naslednjem poglavju 3.2.

3.1.3 Izdelana naprava

Shema celotnega vezja prvega dela se nahaja na sliki 3.4. Poleg prej omenjenih potrebnih delov vezja, pa so del sheme tudi napajalnik, nekaj konektorjev za napajanje in programiranje mikrokrmilnika, testne točke, gumb za resetiranje vezja, nizkopasovni filter za referenčno napetost AD pretvornika in kristal za določanje frekvence delovanja mikrokrmilnika.

Za načrtovanje vezja smo uporabili programsko orodje Eagle podjetja Autodesk, o katerem lahko več preberemo na spletni strani [16]. Načrt vezja, ki ga je izdelala prototipna hiša, si lahko ogledamo na sliki 3.5, na sliki 3.6 pa je prikazano izdelano tiskano vezje.

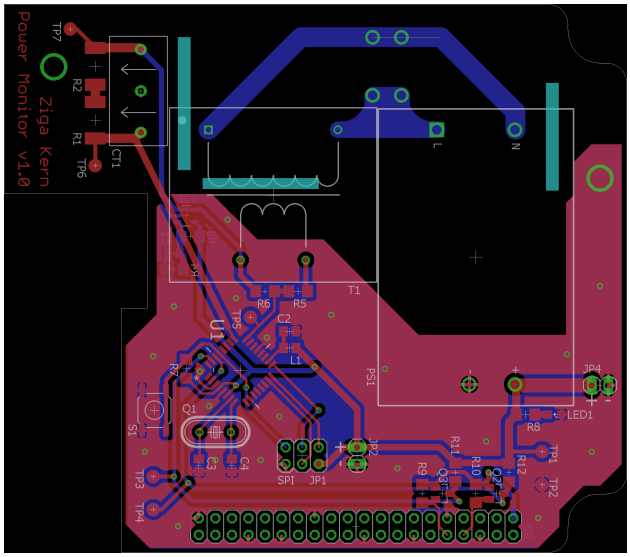
Ohišje, ki vsebuje sestavljeno napravo, sestavljata pokrov, v katerega smo vgradili vtičnico, in škatla z luknjo za napajalni kabel ter 6 distančnikov za pritrditev sestavnih delov naprave. Na sliki 3.7 je prikazan način pritrditve računalnika Raspberry Pi v ohišje, na sliki 3.8 pa je prikazano vstavljeno in privijačeno tiskano vezje. To je bilo pred vstavitvijo sestavljeno s spajkanjem vseh komponent, napajalnega kabla in Micro-B USB kabla, ki služi za napajanje računalnika Raspberry Pi. Na zadnji sliki 3.9 pa je prikazana končana naprava, znotraj katere je napajalni kabel povezan še z vtičnico v pokrovu.



Slika 3.4: Shema celotnega vezja prvega dela naprave.

3.2 Programska oprema

Programsko opremo merilca porabe električne energije lahko opišemo s štirimi glavnimi deli. Prvi kos programske opreme, ki je opisan v podpoglavju 3.2.1, je program nameščen na mikrokrmilniku za branje toka in napetosti, izračun moči ter pošiljanje podatkov preko naprave UART. V podpoglavju 3.2.2 je opisana izbira podatkovne baze za hranjenje podatkov, shema podatkovne baze, opisan način vnosa podatkov in postopek hranjenja ter obdelave teh. V naslednjem poglavju 3.2.3 so opisani algoritmi ujemanja vzorcev, pohitritve



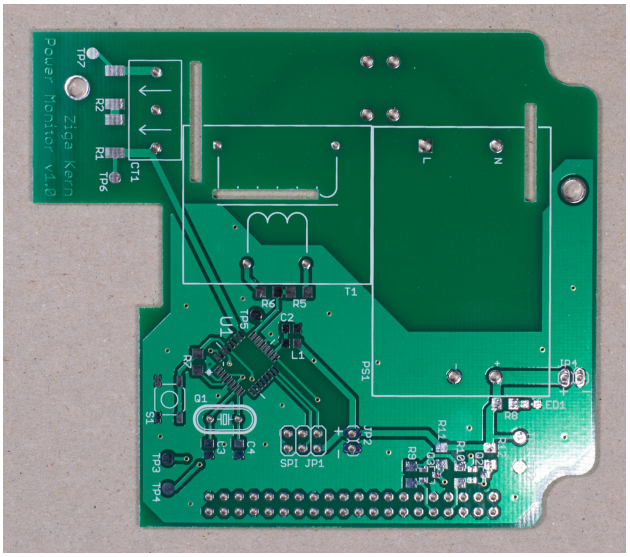
Slika 3.5: Načrt tiskanega vezja.

za izbran algoritem in uporabljen postopek ujemanja vzorcev porabe električne energije v realnem času. V podpoglavju 3.2.4 je predstavljen spletni strežnik, uporabniški vmesnik, sistem za obveščanje o vzorcih in posredovanje podatkov s protokolom MQTT. Poleg te programske opreme smo uporabili še operacijski sistem Raspbian Jessie, ki je naložen na računalnik Raspberry pi 3 in je na voljo preko uradne spletne strani.

3.2.1 Izračun porabe

Izračun moči poteka na mikrokrmilniku ATmega328P. Za pisanje in prevajanje programa smo uporabili razvojno okolje Arduino IDE [10], za programiranje dejanskega mikrokrmilnika pa razvojno okolje Atmel Studio 7 [6]. Programiranje je potekalo s pomočjo programatorja AVR Dragon preko protokola SPI (ang. Serial Peripheral Interface Bus) in priključka JP1 vidnega na tiskanem vezju naprave na sliki 3.6.

Program smo zaradi hitrejšega razvoja in prosto dostopnih knjižnic, napisali v jeziku Arduino. Jezik temelji na jeziku C++ in definira funkcije



Slika 3.6: Prototip tiskanega vezja.

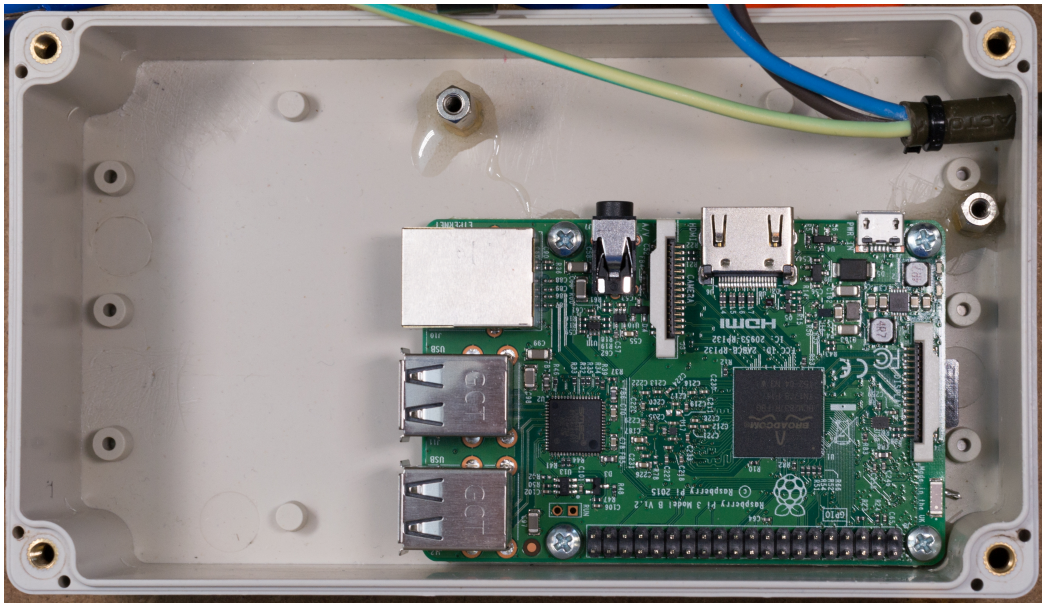
za lahek dostop do strojne opreme, hkrati pa so na voljo tudi vse funkcije knjižnice AVR Libc.

Za izračun moči smo uporabili knjižnico EmonLib [7] in njene funkcije, toda zaradi popolnosti je postopek izračuna še vseeno opisan tukaj.

Meritev napetosti in toka opravimo s pretvorbo analognega signala in branjem rezultatov iz registra ob postavitvi zastavice ADIF, kar storimo s preprostim klicem funkcije *analogRead*. Za merjenje moči izmeničnega toka moramo znotraj ene periode nihaja napetosti opraviti veliko meritev, zato tok in napetost beležimo približno 50-krat na periodo oziroma 2500-krat na sekundo. V primeru omejene procesorske moči ali električne energije ob napajanju z baterijo je frekvenca vzorčenja lahko mnogo manjša, natančnost merjenja pa bi še vedno bila dovolj visoka.

Izračun efektivnega toka, efektivne napetosti, faktorja moči, navidezne in delovne moči nato poteka po postopku opisanem v poglavju 3.1.1.

Pri 50 Hz omrežju se tako podatki o moči izračunajo vsakih 20 ms, zato za manjšo količino podatkov in boljšo natančnost te povprečimo ter pošljemo



Slika 3.7: Računalnik Raspberry Pi pritrjen v ohišje.

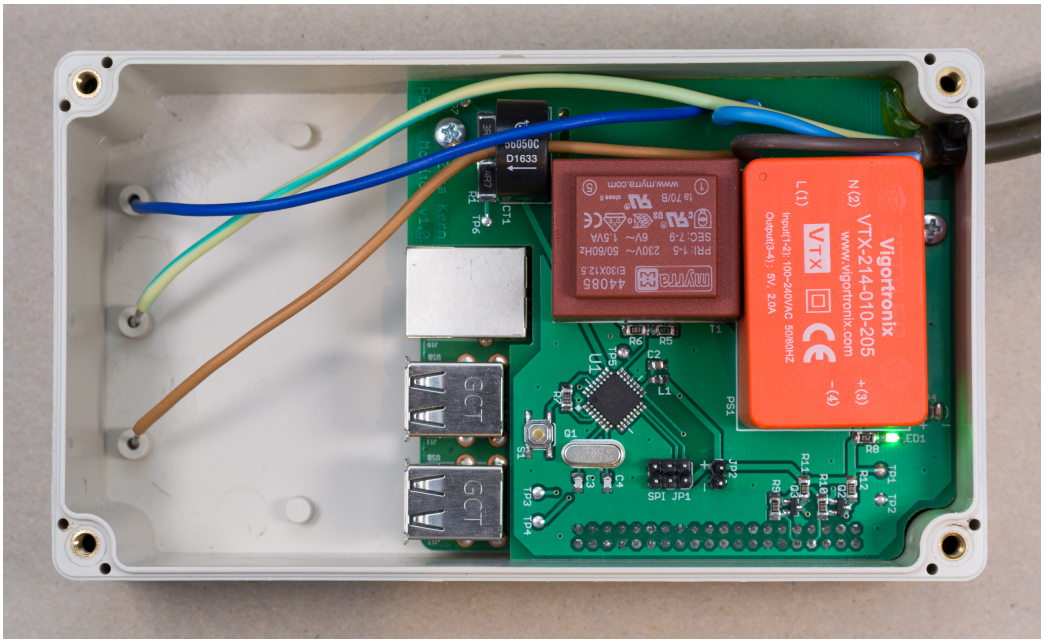
preko naprave UART le približno vsako sekundo.

Univerzalni asinhroni sprejemnik in oddajnik (UART) je naprava za asinhrono serijsko komunikacijo, ki je vgrajena v mnogo integriranih vezij. UART omogoča konfiguracijo hitrosti prenosa in podatkovnega okvirja, ki je ponavadi sestavljen iz začetnega bita, podatkovnih bitov, morebitnega paritetnega in končnega bita.

Hitrost prenosa smo v našem primeru nastavili na 9600 bit/s s pomočjo funkcije *Serial.begin*, podatke pa pošiljamo s funkcijo *Serial.print*. Pri pošiljanju podatkov ne uporabljamo nobenih paritetnih bitov, zaradi nastavljene nizke prenosne hitrosti in kratke povezave med napravama.

UART smo izbrali zaradi podpore tako mikrokrmilnika kot računalnika Raspberry Pi in zato, ker je eden izmed najpreprostejših protokolov za uporabo.

Napravo smo pred uporabo kalibrirali, saj je signal, ki ga vzorčimo z AD pretvornikom, sorazmerno manjši od pravega, kot je zapisano v poglavju



Slika 3.8: Tiskano vezje nameščeno v ohišju in povezano z računalnikom Raspberry Pi.

3.1.1. Naprava je kalibrirana s pomočjo dveh konstant, ki določata razmerje med prebrano in dejansko vrednostjo toka in napetosti.

Konstanti lahko določimo računsko, ali za boljšo natančnost z meritvijo dejanskega toka in napetosti ter izračunom nove konstante. O podrobnostih si lahko preberemo v [14].

3.2.2 Hranjenje podatkov

Če želimo uporabniku prikazati sezonske trende porabe skozi leto, moramo podatke o porabi električne energije v naši napravi hraniti relativno dolg čas. Dolgo hranjenje podatkov pa pomeni tudi veliko količino teh, saj moramo poleg novih podatkov o porabi električne energije, ki jih zabeležimo približno vsako sekundo, hraniti tudi čas, ko so bili ti vneseni v podatkovno bazo.

Za hranjenje naših podatkov potrebujemo podatkovno bazo, ki bo zmožna hitrega poizvedovanja po podatkih za različna časovna obdobja. Najboljša

Slika 3.9: Končana naprava za merjenje porabe električne energije.

rešitev za našo aplikacijo so podatkovne baze TSDB [22].

TSDB so podatkovne baze namenjene za shranjevanje in delo s časovnimi vrstami z velikimi količinami podatkov. TSDB podpirajo množenje, seštevanje ali drugačno združevanje časovnih vrst v realnem času, vključujejo statistične operacije nad časovnimi vrstami, omogočajo zmanjševanje dimenzionalnosti podatkov in podobno. Še ena dobra lastnost TSDB baz, je da omogočajo hiter vnos velike količine podatkov, tudi milijone vnosov na sekundo, kar pomeni, da bi moral za naše potrebe nizko zmogljiv računalnik Raspberry Pi zadostovati. Odločili smo se za podatkovno bazo InfluxDB [13].

Podatke shranjujemo v podatkovno bazo z imenom *power* s petimi meritvami. Meritve pri InfluxDB so podobne tabelam pri SQL podatkovnih bazah. Meritev *power_consumption* se uporablja za shranjevanje vseh podatkov, ki jih Raspberry Pi prejme od mikrokrmilnika. Ti vključujejo delovno in navidezno moč, faktor moči, tok in napetost, poleg teh pa se hrani še čas.

Meritve *day_power*, *week_power* in *year_power* vsebujejo enake podatke, le da so ti obdelani in združeni za daljše časovne intervale, saj s tem zmanjšamo obremenitev podatkovne baze pri poizvedovanju. Zadnja meritev *samples* pa vsebuje podatke o vzorcih, ki se vnesejo, ko uporabnik vzorce porabe električne energije izbere. Podatki o vzorcih so prav tako predhodno obdelani, več o tem v poglavju 3.2.4, vsebujejo pa poleg omenjenih polji tudi polje *id* z imenom vzorca.

Pred začetkom vnašanja podatkov v zgornjo shemo podatkovne baze, smo računalnik Raspberry Pi 3 še skonfigurirali za prejemanje teh preko naprave UART. Po privzetih nastavitvah operacijskega sistema Raspbian Jessie je preko serijskega protokola na voljo konzola, zato smo to v datoteki */etc/inittab* izklopili. Po ponovnem zagonu naj bi bila sedaj povezava vzpostavljena, naprava pa na voljo preko datoteke */dev/serial0*. Podrobneje o vzpostavitvi povezave si lahko preberemo v [4].

Vnos podatkov in poizvedovanje po njih pri podatkovni bazi InfluxDB poteka preko protokola HTTP. Za vnos podatkov smo zato napisali skripto, ki prejema podatke preko naprave */dev/serial0* in pošlje zahtevo HTTP POST podatkovni bazi. Skripta ob njenem začetku preveri obstoj podatkovne baze in jo, v primeru ko ta še ne obstaja, ustvari. Branje podatkov iz naprave nato poteka konstanto. Ko skripta prejme podatke, preveri veljavnost oziroma prisotnost napak in v primeru, da do napak ni prišlo, pošlje zahtevo za vnos podatkovni bazi in počaka na odgovor. V primeru napake pri vnosu, se ta zabeleži v sistemu.

Poleg skripte smo napisali še unit datoteko, ki jo uporablja sistem Systemd. V datoteki so navedene vse potrebne informacije za definicijo storitve, ki je v našem primeru zgornja skripta. Unit datoteko smo shranili v */etc/systemd/system/* ter preko ukaza *systemctl* storitev omogočili. S pomočjo te datoteke Systemd poskrbi za ponovni zagon skripte, če le ta preneha delovati oziroma zagon te ob prižigu naprave. Več o sistemu Systemd, storitvah in pisanju unit datotek si lahko preberemo v poglavjih 9 in

22 uporabniškega priročnika [21].

Kot smo že omenili je dobra lastnost TSDB to, da v primerjavi z drugimi tipi podatkovnih baz podpirajo operacije za delo s časovnimi vrstami. Ena izmed operacij, ki jo InfluxDB podpira je uporaba pravilnika zadrževanja podatkov. S pravili zadrževanja lahko v podatkovno bazo vnašamo podatke, ki se bodo hranili le za določen čas. Ta operacija pride še posebej prav, ko imamo velike količine podatkov ali pa starejših podatkov preprosto ne potrebujemo.

V našem primeru smo ustvarili štiri pravila zadrževanja podatkov za meritve *power_consumtion*, *day_power*, *week_power* in *year_power*. Podatke meritve *power_consumtion* hranimo dva meseca, zaradi možnosti izbire vzorcev porabe za bodoče ujemanje. Podatke meritve *day_power* zadržujemo 24h, podatke meritve *week_power* sedem dni, podatke meritve *year_power* pa 365 dni.

Omejitve zadrževanja podatkov za meritev *samples* ni, posledično se ti podatki ne bodo samostojno izbrisali.

Poleg zgoraj omenjene funkcionalnosti uporabljamo še stalne poizvedbe. Stalne poizvedbe definiramo nad celotno podatkovno bazo, vsebovati pa morajo stavek *INTO* za shranjevanje rezultatov v meritev in *GROUP BY time()* za omejitev časovnega intervala podatkov.

V naši podatkovni bazi smo definirali tri stalne poizvedbe, ki se uporabljajo za obdelavo prispelih podatkov in vstavljanje rezultatov v meritve *day_power*, *week_power* in *year_power*. Prva poizvedba računa povprečje moči v eno minutnih intervalih in rezultate shranjuje v meritev *day_power*. Drugi dve stalni poizvedbi pa računata porabljeno energijo, prva v intervalih dolgih pol ure, druga pa za celoten dan. Rezultati prve se shranjujejo v meritev *week_power*, slednje pa v *year_power*.

S temi stalnimi meritvami zmanjšujemo potrebo po zadrževanju vseh podatkov, zmanjšujemo obremenjenost sistema in pospešujemo prikaz podatkov uporabniku.

Procesiranje in vstavljanje podatkov v meritev *samples* ne poteka preko stalnih poizvedb, temveč se podatki obdelujejo in vstavljajo ob uporabnikovi zahtevi. O postopku izbire vzorcev in shranjevanja bomo govorili v poglavju 3.2.4, o algoritmu za obdelavo teh podatkov za bodoče ujemanje vzorcev pa v poglavju 3.2.3.

3.2.3 Prepoznavanje vzorcev

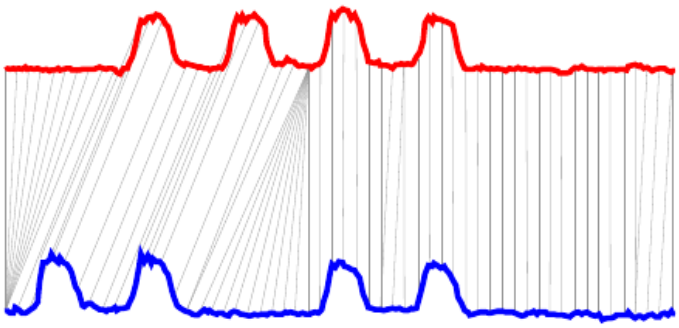
Ena glavnih funkcionalnosti merilca porabe električne energije je obveščanje o vzorcih porabe. Za obveščanje o teh, moramo biti zmožni v realnem času na toku podatkov porabe električne energije prepoznavati vzorce. Prepoznavanje teh poteka s pomočjo algoritmov primerjanja časovnih vrst in mer podobnosti. Ob uspešnem ujemanju vzorcev pa moramo uporabnika o tem tudi obvestiti.

Obstaja mnogo algoritmov in mer za primerjanje vzorcev, najbolj preprosta in najbolj uporabljana mera pa je evklidska razdalja. Slaba lastnost evklidske, manhattanske in drugih razdalj in razlog zakaj te niso primerne je visoka občutljivost na premike in skaliranje tako v horizontalni kot vertikalni osi.

Za skaliranje v časovni osi so po članku [29] primerni algoritmi oziroma mere Levenshteinova razdalja z realno kaznijo (ang. edit distance with real penalty, ERP), dinamično časovno upogibanje (ang. dynamic time warping, DTW) in najdaljši skupni podniz (ang. longest common subsequence, LCSS). Slednja zaradi prisotnosti šuma pri meritvah porabe električne energije za ujemanje vzorcev ni najbolj primerna.

Algoritem DTW oziroma dinamično časovno upogibanje je najbolj primeren za primerjanje časovnih vrst, pri katerih pride do raztezanja, krčenja in premika podatkov v časovni osi. Kljub odpornosti skaliranju v horizontalni osi, je še vedno občutljiv na skaliranje v vertikalni osi, zato je mnogokrat potrebno podatke predhodno normalizirati.

Algoritem DTW računa minimalno razdaljo med dvema časovnima vrstama s primerjanjem točk vrst, ki se najbolj ujemajo, kot je prikazano na sliki 3.10. Bolj natančno povedano, algoritem DTW išče najbolj optimalno pot, pri kateri je razdalja med vrstama minimalna, skozi matriko velikosti $n \times m$, kjer sta n in m dolžini časovnih vrst ter sta v primerjavi z evklidsko razdaljo lahko različni. Bolj podrobno razlago si lahko preberemo v članku [18], povedali pa bi, da je časovna zahtevnost algoritma DTW v splošnem enaka $O(nm)$, vendar za algoritem obstaja več pospešitev.



Slika 3.10: Primerjava dveh časovnih vrst z DTW. Slika iz [2].

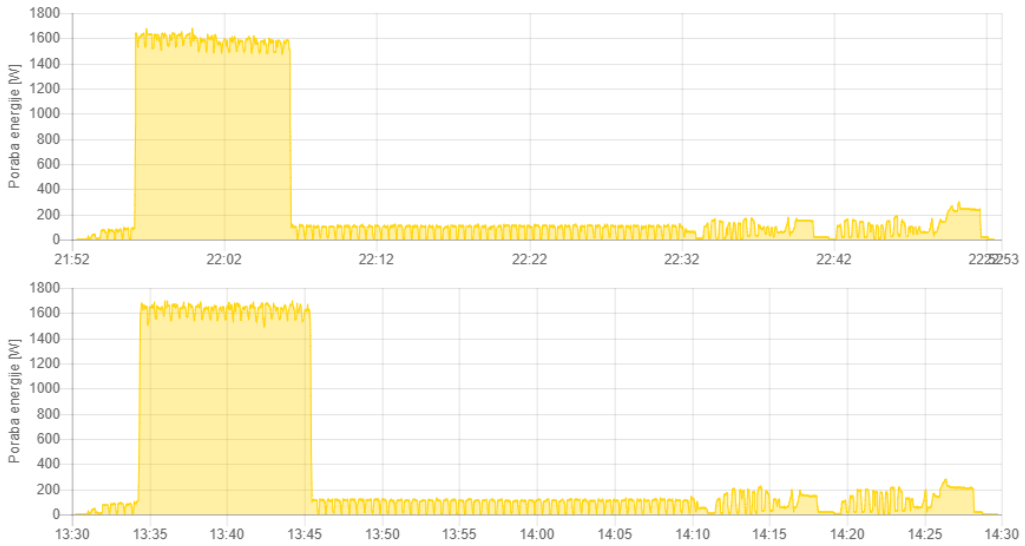
Algoritem za računanje razdalje med časovnima vrstama ERP, podobno kot algoritem DTW, išče najkrajšo razdaljo. Toda ta v primerjavi z algoritomom DTW dopušča izpuščanje točk vrste, ki se ne ujemajo s točkami druge. Izpuščanje delov vzorca porabe v našem primeru ni zaželeno, saj poraba velike večine naprav sledi v naprej določenim vzorcem in bi izpuščanju sledilo le napačno ujemanje vzorcev.

Poleg ujemanja vzorcev s predlogami, kot ga uporabljamo v tem diplomskem delu, obstajajo tudi pristopi za iskanje vzorcev z uporabo pravil. Pri iskanju vzorcev s pravili se išče vzorce z določeno obliko časovne vrste. V našem primeru si želimo visoko natančnost prepoznavanja vzorcev porabe električne energije in razlikovanje teh, tudi če so si ti podobni, zato se za to

metodo nismo odločili, primerjavo metod iskanja si pa lahko ogledamo v [19].

Najprimernejši algoritem za primerjanje podatkov o porabi električne energije je algoritem DTW, predvsem zaradi fleksibilnosti v časovni osi, saj čas porabe energije mnogokrat ni konstanten in je odvisen od zunanjih dejavnikov. Kot primer lahko vzamemo čase delovanja mikrovalovne pečice, televizije in klimatske naprave, ki so odvisni od uporabnikovih nastavitev in zunanjih dejavnikov. Primer vpliva zunanjega dejavnika vidimo na sliki 3.11, kjer je čas gretja vode ob začetku pranja pralnega stroja odvisen od začetne temperature vode.

Prav tako občutljivost algoritma na vertikalno skaliranje podatkov ne vpliva na ujemanje vzorcev porabe električne energije, saj je ta za naprave v enakih pogojih konstantna in posledično normalizacija ni potrebna.



Slika 3.11: Primerjava porabe električne energije pralnega stroja z enakimi nastavitvami.

Za ujemanje velikega števila daljših vzorcev z veliko podatki v realnem času je uporaba osnovnega algoritma DTW brez pohitritev prepočasna. Prva možnost za pohitritev procesiranja je zmanjševanje količine podatkov. Za

zmanjševanje količine podatkov lahko uporabimo eno izmed mnogih tehnik za zmanjševanje dimenzionalnosti podatkov, ki so opisane in podrobno predstavljene v [18].

Za zmanjševanje količine podatkov smo se odločili za PAA oziroma aproksimirano kosovno agregiranje podatkov. S PAA časovno vrsto dolžine N predstavimo z vektorjem

$$X = (x_1, x_2, ..., x_n),\ x_i = \frac{n}{N} \sum_{j=\frac{N}{n}(i-1)+1}^{\frac{N}{n}i} x_j \tag{3.10}$$

dolžine n, kjer je n poljubno število.

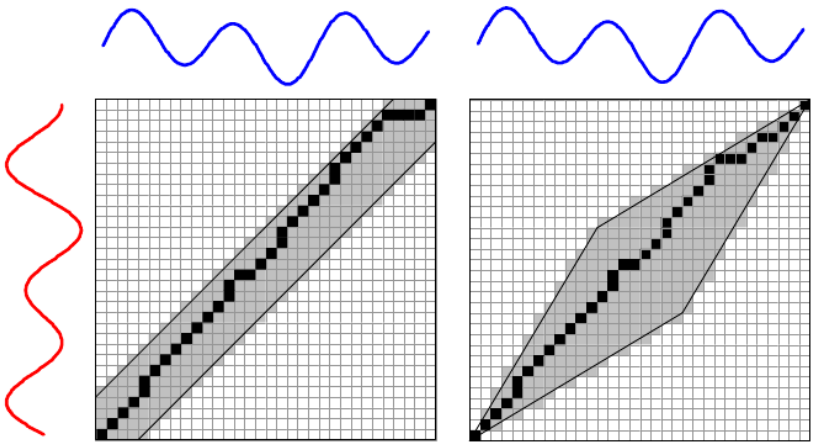
S tem postopkom časovno vrsto predstavimo kot n povprečij. Vsako povprečje pa je izračunano iz enako dolgih časovnih intervalov prvotne časovne vrste, ki smo jih dobili z razdelitvijo vrste na n kosov.

PAA smo izbrali zaradi preprostosti računanja, njegove računske zahtevnosti $O(n)$ in posledično hitrosti. Prednost izbire PAA pred ostalimi tehnikami zmanjševanja dimenzionalnosti podatkov, je tudi ta, da je implementiran v izbrani podatkovni bazi InfluxDB.

Poleg zmanjšanja količine podatkov, lahko DTW pohitrimo z omejitvijo prostora raziskovanja. Najbolj preprosta in efektivna omejitev prostora raziskovanje je s pasom Sakoe-Chiba ali s paralelogramom Itakura, ki sta prikazana na sliki 3.12. Z uporabo enega izmed njiju omejimo horizontalno prilagodljivost algoritma DTW, saj se točke ene časovne vrste lahko ujemajo le z točkami druge, ki niso oddaljene za več kot je širina raziskovalnega pasu oziroma paralelograma v tisti vrsti ali stolpcu matrike.

Pri primeru porabe električne energije je od teh dveh bolj smiselna uporaba pasu Sakoe-Chiba, saj se pri zamiku celotnega vzorca porabe zadnji in prvi del ne ujemata z drugim vzorcem. V primeru uporabe Itakura paralelograma se z zamikom celotnega vzorca tako lahko izloči potencialne rešitve.

Algoritem DTW lahko po omejitvi raziskovalnega prostora pohitrimo še z zgodnjim opuščanjem rešitev z uporabo spodnje meje. Za spodnjo mejo



Slika 3.12: Sakoe-Chiba pas na levi in Itakura paralelogram na desni. Slika vzeta iz [2].

pa uporabljamo algoritem LB_Improved [27], ki je izboljšava pogosto uporabljene spodnje meje LB_Keogh [25].

Program za prepoznavanje vzorcev v porabi električne energije smo napisali v razvojnem okolju Visual Studio 2015 v jeziku C++. Implementacijo algoritma DTW s spodnjo mejo LB_Improved pa je priskrbela knjižnica LBImproved [8].

Program ob zagonu iz meritve *samples* podatkovne baze InfluxDB pridobi vzorce porabe električne energije, ki so bili ob vnosu obdelani z algoritmom PAA. Za vsak vzorec se nato izračuna porabljena energija in spodnja meja z algoritmom LB_Improved, nato pa se začne konstantno primerjanje vzorcev s trenutno porabo.

Program iz podatkovne baze pridobi trenutne podatke porabe električne energije, katere podatkovna baza obdela z zgoraj opisanim algoritmom PAA. Nato za vsak vzorec program preveri če porabljena energija zadnjega obdobja ustreza energiji vzorca, ki je izračunana ob začetku programa. V primeru, da si ti nista podobni, program nadaljuje na naslednji vzorec, saj s tem dosežemo še hitrejše preverjanje, v nasprotnem primeru, pa se ujemanje

trenutne porabe z vzorcem preveri z algoritmom DTW.

Ko je najdeno dobro ujemanje vzorca s trenutno porabo program preverjanje začasno ustavi in pošlje zahtevo za obvestilo skupaj s porabo ter imenom zaznanega vzorca spletnemu strežniku, ki nato primerno obvesti uporabnika naprave, več o tem pa v naslednjem poglavju 3.2.4.

3.2.4 Uporabniški vmesnik

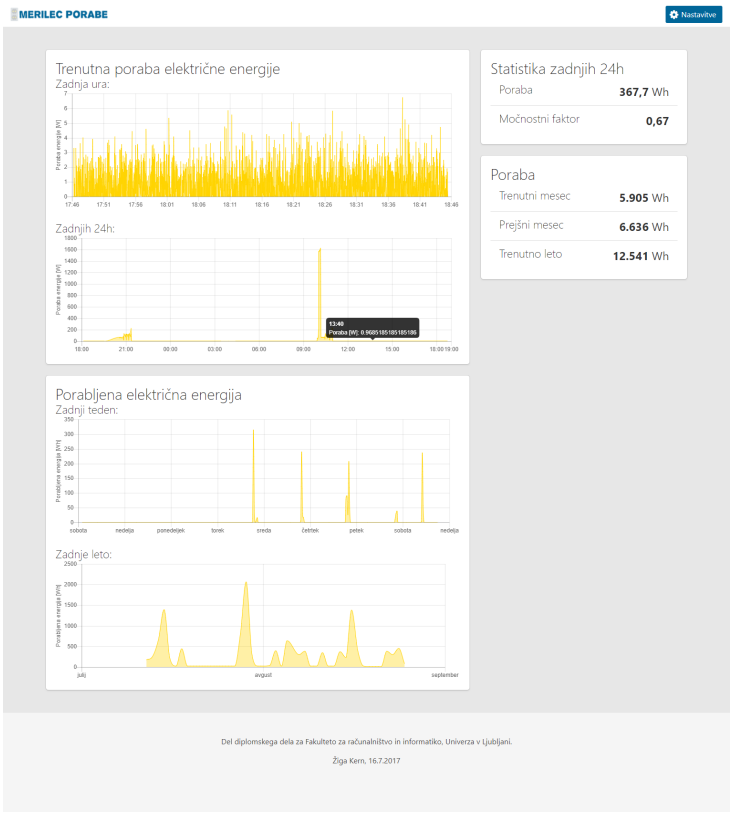
Uporabniški vmesnik merilca porabe električne energije je spletna stran, kjer opravljamo z napravo in imamo pregled nad porabo električne energije. Spletno stran smo namesto aplikacije za mobilne naprave izbrali zaradi dostopnosti iz večjega števila naprav in enotnega vmesnika.

Za spletni strežnik smo izbrali Nginx in izvajalno okolje Node.js skupaj z programskim ogrodjem Express. Nginx, ki deluje kot namestniški strežnik za Node.js, uporabljamo za strežbo statičnih datotek in za strežbo spletne strani preko HTTPS, kar je potrebno za spletna obvestila o zaznanih vzorcih, o katerem govorimo v naslednjih odstavkih. Za razvoj spletne strani smo uporabili razvojno okolje JetBrains WebStorm.

Skupaj z Node.js uporabljamo še pakete Chart.js za generacijo in prikazovanje grafov na spletni strani, Request za pošiljanje HTTP zahtev podatkovni bazi, MQTT za pošiljanje podatkov drugim napravam in Web-push za pošiljanje spletnih obvestil. Poleg teh uporabljamo še CSS ogrodje Bulma in stroj za predloge (ang. template engine) Handlebars.

Uporabniški vmesnik je sestavljen iz dveh delov, iz pregleda porabe električne energije in nastavitev, s katerimi opravljamo z merilcem porabe.

Na domači strani iz slike 3.13, kjer se nahaja pregled porabe, so prikazani štirje grafi in nekaj statistike. Prvi graf prikazuje moč in po izbiri še faktor moči naprave za zadnjo uro, drugi pa moč za zadnjih štiriindvajset ur. Na tretjem in četrtem grafu se nahaja porabljena električna energija v vatnih urah za pretekli teden in leto, statistika pa prikazuje porabljeno energijo za zadnjih štiriindvajsetih ur, za trenutni in pretekli mesec ter za tekoče leto.



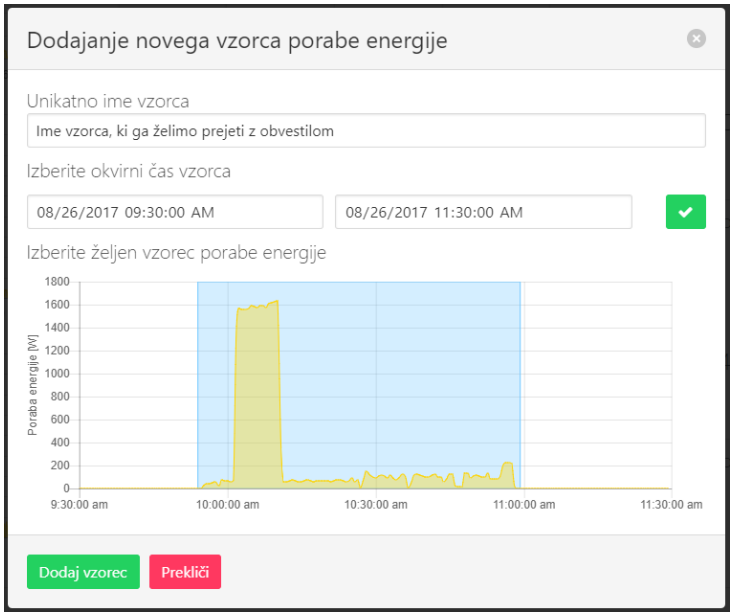
Slika 3.13: Domača stran, ki prikazuje podatke o porabi električne energije.

Poleg teh podatkov je prikazan še močnostni faktor za zadnjih štiriindvajsetih ur.

Na strani z nastavitvami, prikazani na sliki 3.15, so na voljo nastavitve za videz grafov, hitrost osveževanja podatkov na domači strani, nastavitve za obveščanje o vzorcih in nastavitve za pošiljanje podatkov drugim napravam preko protokola MQTT.

Za obveščanje smo na tej strani implementirali dodajanje vzorcev porabe električne energije in urejanje teh. Vzorce se doda z vnosom unikatnega imena vzorca in izbiro časovnega intervala porabe električne energije o katerem bi bili radi obveščeni. Za izbiro časovnega intervala je bil po meri narejen nov tip grafa paketa Chart.js, na katerem lahko označujemo časovni interval,

kot je prikazano na sliki 3.14. Ob dodajanju novega vzorca, se podatki obdelajo z algoritmom PAA, opisanem v prejšnjem poglavju 3.2.3, in shranijo v meritev *samples* podatkovne baze *power*.

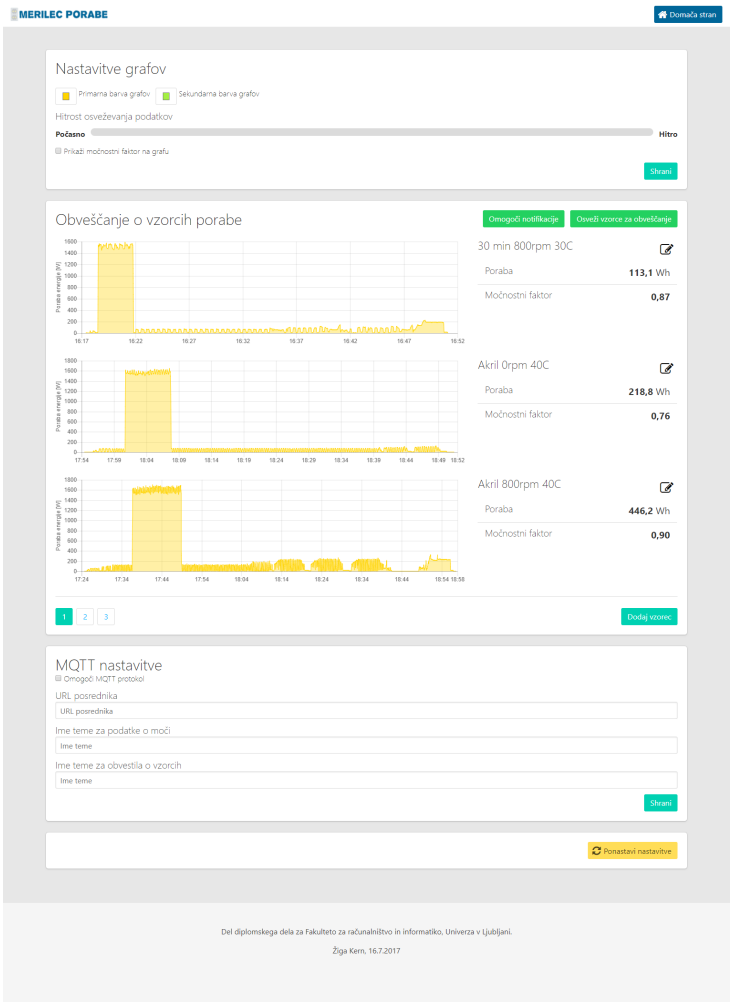


Slika 3.14: Pojavno okno za dodajanje novega vzorca porabe električne energije.

Za obveščanje uporabnika ob zaznanem vzorcu uporabljamo spletna push obvestila (ang. web push notifications). Za pošiljanje spletnih obvestil je zahtevana uporaba HTTPS, na strani odjemalca pa je potrebno registrirati storitvenega delavca (ang. service worker) in zaprositi za dovoljenje uporabnika. Storitveni delavec je koda javascript, ki se lahko izvaja v ozadju tudi ko spletna stran ni odprta in prikazuje sporočila, ki jih ta prejme. Sporočila pa na zahtevo programa za prepoznavanje vzorcev pošilja spletni strežnik, podroben opis pošiljanja teh najdemo v [17].

Implementiran je bil tudi MQTT odjemalec, ki omogoča pošiljanje podatkov o porabi električne energije naprave in zaznanih vzorcih drugim napra-

vam in storitvam. Na strani z nastavitvami je mogoče nastaviti naslov URL posrednika MQTT in imena tem, preko katerih se nato podatki pošiljajo.



Slika 3.15: Spletna stran z nastavitvami naprave.

Poglavje 4

Rezultati

V okviru diplomskega dela smo uspešno izdelali prototip merilca porabe električne energije, ki ga lahko uporabljamo za nadgradnjo različnih naprav. Merilec, ki ga vidimo na sliki 3.9, je preprost za uporabo. Namestimo ga podobno kot pametno vtičnico, tako da napravo vklopimo v vtičnico merilca, napajalni kabel merilca pa v vtičnico, kamor je bila pred tem vključena naprava. Merilec porabe električne energije se nato samostojno prižge in poveže na brezžično omrežje.

Z nadgradnjo lahko napravo sedaj preko protokola MQTT povezujemo z kontrolerji, ki smo jih omenili v poglavju 2, ali v druge sisteme. Žal pa nam prototip merilca porabe ne omogoča upravljanja naprave, temveč le pregled delovanja te.

Testiranje merilca porabe električne energije je potekalo v štiri članskem gospodinjstvu na pralnem stroju Candy GV 138TWHC3/1-S. Napravo smo uporabljali približno mesec dni, v tem času je bilo izvedenih okrog 30 pranj oblačil in porabljeno 13 kWh električne energije.

S pregledom porabe na domači strani, kot je prikazana na sliki 3.13, lahko ugotovimo vzorce porabe v daljših časovnih obdobjih in glede na napravo s pomočjo podatkov zmanjšamo porabo energije. V primeru pralnega stroja smo ugotovili, da se naprava med tednom uporablja kvečjemu enkrat v popoldanskih urah in bi z zamikom časa pranja v čas cenejše tarife lahko

privarčevali.

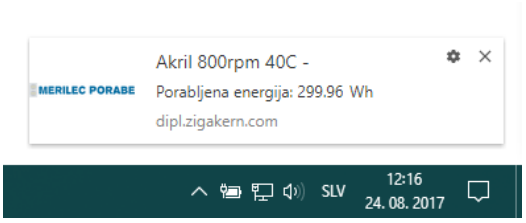
V obdobju testiranja je eno pranje oblačil v pralnem stroji povprečno uporabilo približno 400 Wh energije, kar je pri ceni električne energije v času testiranja okrog 0,04 EUR na pranje, posledično ima izbira programa pranja relativno majhen vpliv na ceno. Največji delež cene pranja prispevata detergent in mehčalec, ki ga uporabljamo pri pranju in ne cena porabljene vode ali električne energije.

Na začetku obdobja testiranja smo na strani z nastavitvami, ki je prikazana na sliki 3.15, vnesli devet vzorcev porabe električne energije najbolj pogostih programov pranja. Glede na čas primerjanja vzorcev računalnika Raspberry Pi 3, sklepamo, da bi za preverjanje prisotnosti vzorcev v realnem času lahko vnesli od 30 do 40 enournih vzorcev porabe električne energije ali manjše število daljših vzorcev.

V primeru ko je število vzorcev preveliko ali so vzorci predolgi, se preverjanje prisotnosti teh v porabi ne izvaja za vsak nov vnos podatkov v podatkovno bazo, temveč le periodično. Kljub temu, zaznavanje do neke mere še vedno deluje zaradi izbire algoritma, ki je odporen na časovne zamike. Slabost je le, da se nekoliko zamakne čas, ko je vzorec zaznan, pri dovolj velikem številu vzorcev pa zanesljivo zaznavanje vzorcev ne bi bilo več možno.

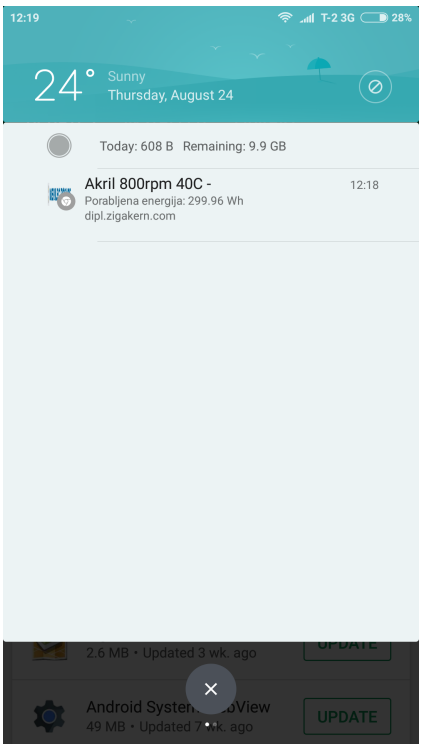
Zaznavanje vzorcev porabe električne energije je bilo v obdobju testiranja uspešno. Vzorci porabe programov, ki so bili predhodno dodani v napravi, so bili v naslednjih pranjih zaznani v roku od nekaj minut pred koncem programa, do malo po njem. Ob uspešno zaznanem vzorcu porabe smo preko brskalnika na telefonu in stacionarnem računalniku prejeli obvestilo, kot je prikazano na sliki 4.1 in 4.2.

Pri obveščanju in zaznavanju vzorcev v porabi električne energije pa smo našli tudi nekaj pomanjkljivosti. Pomanjkljivost obveščanja uporabnika preko spletnih push obvestil je, da je za prejem obvestila potrebno imeti odprt brskalnik v ospredju ali v ozadju. V primeru ko tega zapremo, obvestilo



Slika 4.1: Primer obvestila o zaznanem vzorcu na stacionarnem računalniku.

prejmemo šele ob naslednjem zagonu. Pomanjkljivost pri zaznavanju vzorcev porabe pa je lažno zaznavanje vzorca porabe, ki je podoben delu drugega, saj v tem primeru prejmemo obvestilo za napačno zaznan vzorec, za tem pa še za pravega. V obdobju testiranja je naprava tako v 10,5 % primerov (2x) predhodno zaznala napačen vzorec, v 5,3 % primerov (1x) vzorca ni zaznala zaradi majhne količine oblačil in posledično manjše porabe energije, v 84,2 % primerov (16x), pa je bil vzorec pravilno zaznan.



Slika 4.2: Primer obvestila o zaznanem vzorcu na mobilnem telefonu.

Poglavje 5

Zaključek

V diplomskem delu smo predstavili področje IoT skupaj s trenutno ponudbo naprav in storitev. Preučili smo možnosti nadgradnje obstoječih naprav in ugotovili, da nekaj možnosti za nadgradnjo že obstaja in da so te predvsem odvisne od vrste naprave. Nato smo se osredotočili na izdelavo prototipa naprave, ki nam omogoča nadgradnjo obstoječih porabnikov energije, kot so pralni, sušilni, pomivalni stroji, pečice in druge naprave.

Opisali smo način merjenja porabe električne energije naprav in izdelali prototip merilca porabe. Tega smo izdelali s pomočjo računalnika Raspberry Pi in po meri narejenega tiskanega vezja za merjenje toka ter napetosti, pri katerem bi bilo v prihodnosti potrebno dodati varovalko in prenapetostno zaščito.

Napisali smo program za izračun moči iz toka in napetosti ter sprogramirali mikrokrmilnik. Vzpostavili smo povezavo med mikrokrmilnikom in računalikom Raspberry Pi ter implementirali shranjevanje podatkov o porabi v podatkovno bazo TSDB.

Izdelali smo spletno stran za uporabniški vmesnik. Implementirali smo izbiro vzorcev, predstavili in izbrali algoritem DTW za primerjanje časovnih vrst, predstavili možnosti za pohitritev izbranega algoritma, implementirali zaznavanje vzorcev v porabi električne energije in implementirali sistem za obveščanje o najdenih vzorcih preko spletnih obvestil.

Na koncu smo dodali možnost posredovanja podatkov preko protokola MQTT za uporabo naprave v obstoječih sistemih IoT naprav, nato pa smo napravo še testirali v gospodinjstvu. Iz testiranja smo ugotovili, da je detekcija vzorcev uspešna. Zaznavanje vzorcev bi v nadaljevanju lahko nadgradili tudi s pravili, s katerimi bi se izognili vnašanju vseh vzorcev porabe.

Poleg tega smo ugotovili, da bi bilo potrebno implementirati vnos podatkov za povezovanje v brezžično omrežje, za izboljšavo merilca porabe pa bi lahko vključili rele za vklop in izklop naprave ter uporabili sistem obveščanja, ki ne zahteva odprtega brskalnika.

Literatura

[1] 50/60hz current sensor – cs60-010. Dosegljivo: http://www.farnell.com/datasheets/1870392.pdf?_ga=2.235441092.476209616.1502883197-501887449.1502883197, 2005. [Dostopano: 16. 8. 2017].

[2] Converting images into time series for data mining. Dosegljivo: https://izbicki.me/blog/converting-images-into-time-series-for-data-mining.html, 2011. [Dostopano: 19. 8. 2017].

[3] Irkit opensource infrared remote controller. Dosegljivo: http://getirkit.com/en/, 2014. [Dostopano: 13. 8. 2017].

[4] Serial comms over hardware uart for johnny-five between arduino and a raspberry pi. Dosegljivo: https://gist.github.com/ajfisher/8880882, 2014. [Dostopano: 17. 8. 2017].

[5] Atmega328p: 8-bit avr microcontroller with 32k bytes in-system programmable flash. Dosegljivo: http://www.atmel.com/images/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf, 2015. [Dostopano: 16. 8. 2017].

[6] Atmel studio 7 - easier to use and more powerful than ever. Dosegljivo: http://www.atmel.com/microsite/atmel-studio/, 2016. [Dostopano: 17. 8. 2017].

[7] Emonlib: Arduino energy monitoring library - compatible with arduino 1.0. Dosegljivo: https://github.com/openenergymonitor/EmonLib, 2016. [Dostopano: 17. 8. 2017].

[8] Lbimproved c++ library. Dosegljivo: https://github.com/lemire/lbimproved, 2016. [Dostopano: 23. 8. 2017].

[9] Amazon web services iot. Dosegljivo: https://aws.amazon.com/iot/, 2017. [Dostopano: 4. 6. 2017].

[10] Arduino - software. Dosegljivo: https://www.arduino.cc/en/Main/Software, 2017. [Dostopano: 17. 8. 2017].

[11] Azure iot suite. Dosegljivo: https://www.microsoft.com/en-us/internet-of-things/azure-iot-suite, 2017. [Dostopano: 4. 6. 2017].

[12] Google cloud iot. Dosegljivo: https://cloud.google.com/solutions/iot/, 2017. [Dostopano: 4. 6. 2017].

[13] Influxdata (influxdb) — time series database monitoring & analytics. Dosegljivo: https://www.influxdata.com/, 2017. [Dostopano: 17. 8. 2017].

[14] Learn — openenergymonitor. Dosegljivo: https://learn.openenergymonitor.org/, 2017. [Dostopano: 4. 5. 2017].

[15] Official raspberry pi page. Dosegljivo: https://www.raspberrypi.org, 2017. [Dostopano: 2. 6. 2017].

[16] Pcb design and schematic software — eagle — autodesk. Dosegljivo: https://www.autodesk.com/products/eagle/overview, 2017. [Dostopano: 16. 8. 2017].

[17] Web push notifications: Timely, relevant, and precise. Dosegljivo: https://developers.google.com/web/fundamentals/engage-and-retain/push-notifications/, 2017. [Dostopano: 24. 8. 2017].

[18] Carmelo Cassisi, Placido Montalto, Marco Aliotta, Andrea Cannata, and Alfredo Pulvirenti. Similarity measures and dimensionality reduction techniques for time series data mining. In *Advances in data mining knowledge discovery and applications*, pages 342–376. InTech, 2012.

[19] Tak chung Fu; Fu-lai Chung; Robert Luk; Chak-man Ng. Stock time series pattern matching: Template-based vs. rule-based approaches. *Engineering Applications of Artificial Intelligence*, 20(3), 2007.

[20] RS Components. 11 Internet of Things (IoT) Protocols You Need to Know About. Dosegljivo: `https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about`, 2015. [Dostopano 31. 6. 2017].

[21] Marie Doleželová, Maxim Svistunov, Stephen Wadeley, Tomáš Čapek, Jaromír Hradílek, Douglas Silas, Jana Heves, Petr Kovář, Peter Ondrejka, Petr Bokoč, Martin Prpič, Eliška Slobodová, Eva Kopalová, Miroslav Svoboda, David O'Brien, Michael Hideo, Don Domingo, and John Ha. *Red Hat Enterprise Linux 7 System Administrator's Guide*. Red Hat Inc, 2017.

[22] Ted Dunning and Ellen Friedman. *Time Series Databases: New Ways to Store and Access Data*. O'Reilly Media, 2014.

[23] Keith D. Foote. A brief history of the internet of things. Dosegljivo: `http://www.dataversity.net/brief-history-internet-things/`, 2016. [Dostopano: 10. 8. 2017].

[24] Stan Gibilisco. *Electricity and Electronics, 4th Edition*. McGraw-Hill/TAB Electronics, 2006.

[25] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[26] Sheetal Kumbhar. The number of smart homes in Europe and North America tops 17.9 million in 2015. Dosegljivo: https://www.iot-now.com/2016/05/31/47936-the-number-of-smart-homes-in-europe-and-north-america-tops-17-9-million-in-2015/, 2016. [Dostopano 31. 5. 2017].

[27] Daniel Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern Recogn.*, 42(9):2169–2180, 2009.

[28] Donald Norris. *The Internet of things: do-it-yourself projects with Arduino, Raspberry Pi, and BeagleBone Black*. McGraw-Hill Education TAB, 2015.

[29] D Rajalakshmi and K Dinakaran. A survey on effective pattern matching in uncertain time series stream data. *Asian Journal of Applied Sciences*, 8(3):217–226, 2015.

[30] John Romkey. Toast of the iot: The 1990 interop internet toaster. *IEEE Consumer Electronics Magazine*, 6(1):116–119, 2017.

[31] Paul Scherz and Simon Monk. *Practical Electronics for Inventors, 4th Edition*. McGraw-Hill Education TAB, 2016.

[32] Marco Schwartz. *Internet of Things with the Arduino Yun*. Packt Publishing, 2014.

[33] Google Trends. Google trends: internet of things. Dosegljivo: https://trends.google.com/trends/explore?date=all&q=%22internet%20of%20things%22&hl=en-US. [Dostopano: 11. 8. 2017].

[34] Rob van der Meulen. Gartner says 8.4 billion connected "things"will be in use in 2017, up 31 percent from 2016. Dosegljivo: https://www.gartner.com/newsroom/id/3598917, 2017. [Dostopano 31. 5. 2017].

[35] Mark van Rijmenam. Where does the internet of things come from? Dosegljivo: https://datafloq.com/read/where-does-the-internet-of-things-come-from/524, 2014. [Dostopano: 10. 8. 2017].